Perspective Correcting Visual Odometry for Agile MAVs using a Pixel Processor Array

Colin Greatwood[1], Laurie Bose[1], Thomas Richardson[1], Walterio Mayol-Cuevas[1]
Jianing Chen[2], Stephen J. Carey[2] and Piotr Dudek[2]

***Abstract*— This paper presents a visual odometry approach using a Pixel Processor Array (PPA) camera, specifically, the SCAMP-5 vision chip. In this device, each pixel is capable of storing data and performing computation, enabling a variety of computer vision tasks to be carried out directly upon the sensor itself. In this work the PPA performs HDR edge detection, perspective correction and image alignment based odometry, allowing the position and heading of a MAV to be tracked at several hundred frames per second.**

We evaluate our PPA based approach by direct comparison with a motion capture system for a variety of trajectories. These include rapid accelerations that would incur significant motion blur at low frame rates, and lighting conditions that would typically lead to under or over exposure of image detail. Such challenging conditions would often lead to unusable images when relying on traditional image sensors.

I. INTRODUCTION

In order to successfully navigate, autonomous Micro Air Vehicles (MAVs) require the ability to sense their position. Typically GPS is used, but in certain situations such as indoor flight the vehicle must fall back to relying upon its onboard sensors. Onboard cameras and computer vision are commonly used to estimate robot motion, ranging from lateral motion sensing to full 6DOF Simultaneous Localisation And Mapping (SLAM). Several studies have demonstrated SLAM utilising FPGAs, powerful single board computers, stereo cameras and RGB-D cameras e.g. [1]–[5]. In this work we do not attempt to reconstruct a detailed map of the world and rather focus on high frame rate estimation of the platform's motion. For a number of MAV applications which require processing to be fully conducted onboard the vehicle, simply estimating the platform's motion and location relative to a starting position is sufficient and more achievable than mapping the environment. Visual odometry is one such approach.

This paper presents a visual odometry strategy using a Pixel Processor Array (PPA) camera to estimate a quadrotor's velocity, heading and position at up to 500 Hz. These quantities are required for the vehicle to perform almost any form of autonomous flight, thus the visual odometry is vital in enabling autonomy in GPS denied environments.

A MAV's lateral movement could be measured using a dedicated downward facing device such as the PX4FLOW [6], which similar to a mouse sensor, returns lateral velocity at about 250 Hz. However, these sensors do not detect changes in the vehicle's yaw, instead relying on the autopilot's digital compass to aid position determination.

Traditional camera sensors have been used for MAV localisation, but typically require additional computing power to process the data. For example, the downward facing camera on the AR.Drone [7] can provide images for processing on the onboard computer at up to 60 frames per second for optical flow, furthermore template matching at a lower frame rate is used to reduce drift. Point features may also be extracted from the image to estimate movement especially when fused with the IMU [8]. The drawback of these approaches, however, is that the computational power required to perform these evaluations leads to either low frame rates or higher mass MAVs.

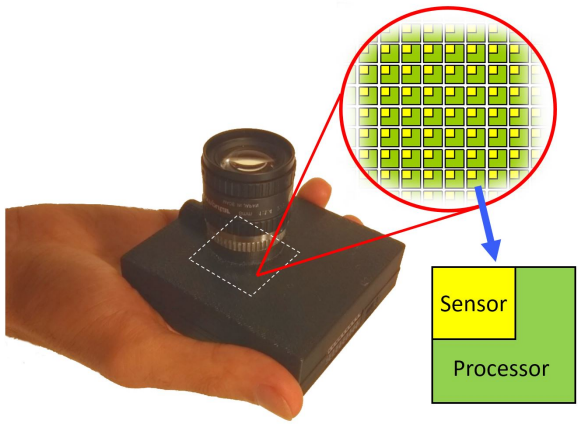

Fig. 1: SCAMP-5

In contrast to conventional image sensors, the SCAMP-5 PPA (Fig 1) used in this work can perform high frame rate, low latency, visual odometry motion estimation with only trivial onboard processing requirements. PPA sensors consist of a parallel array of processing elements, each of which features light capture, processing and storage capabilities allowing for various image processing tasks to be efficiently performed directly on the sensor [9], [10]. Crucially, the PPA only needs to output specific information such as the estimated egomotion variables, rather than having to output entire images. This vastly reduces the bandwidth required per frame in communication between the sensor and onboard computer, enabling high frame rate low power operation by the entire system.

In this work the motion of the quadrotor MAV is estimated

*This work was conducted at the Bristol Robotics Laboratory
[1]Faculty of Engineering, Aerospace and Computer Science, University of Bristol, Bristol, England colin.greatwood@bristol.ac.uk
[2]School of Electrical and Electronic Engineering, The University of Manchester, Manchester, England p.dudek@manchester.ac.uk

from the rate at which the scene moves across the images captured by the PPA sensor. This scene tracking is achieved by an image alignment process, building upon previous work in [11], where the current image is aligned against a previous acquired key-frame, in a manner exploiting the parallel nature of the PPA architecture. The output of this scale-less visual odometry method is converted into position and velocity using the MAV's height sensor, as discussed in Section III-A. Additionally this work presents a novel PPA perspective correction algorithm, in which images of a surface are warped to consistently appear as if acquired directly facing said surface. This ensures the visual odometry receives consistent images of the ground plane, despite any rolling and pitching motions of the MAV. This perspective correction method is introduced in Section III-B. Another benefit of the PPA is the ability to control the exposure time for data captured at each pixel, allowing for various High Dynamic Range (HDR) capabilities. Section III-C presents a new method for capturing HDR edge images, which are then used by the visual odometry algorithm. The integration of the resulting visual odometry data is described in Section III-D, followed by flight results in Section IV.

II. HARDWARE

A. SCAMP-5

The visual odometry presented in this paper was conducted entirely upon a SCAMP-5 PPA [10], [12], [13]. No other device was used in processing visual data. The SCAMP-5 integrated circuit features an array of 256×256 processing elements (PEs), each capable of light capture, storage and processing of visual data - effectively putting a small "microprocessor" inside every pixel of the sensor array. The pixels feature a photosensor, local analog and digital memory, and the ability to perform logic and arithmetic operations. Each PE may also communicate with each of its four neighboring elements in the array, making it possible to transfer register data across PEs. A programmable controller chip issues identical instructions to each PE of the array, which then all perform said instruction simultaneously, in parallel. In this way processing follows the standard single instruction multiple data (SIMD) approach and allows for efficient parallel processing. Vision algorithms can then be executed directly in the pixel array, without ever transmitting the images out of the sensor device.

By only sending the meaningful data such as the values relating to detected camera ego-motion, there is a significant decrease in the bandwidth and hence power required during operation. This approach allows many visual tasks to be conducted at very high frame-rates (such as at 100,000 fps in [10]), something typically not possible using the standard visual processing pipeline. SCAMP-5 is also low power, requiring below 2 Watts, which compares well with GPU-based approaches that while parallel, require 10s-100s of Watts. The SCAMP inside the 3D printed box used here weighs 100 grams and measures 82x77x23mm excluding the lens.

B. Flight Hardware

A custom quadrotor, show in Figure 2, was designed and built to carry the SCAMP vision sensor facing either forwards or downwards. The quadrotor weighs 1kg with SCAMP installed and measures 400mm diagonally between rotors. In the work presented here the sensor was mounted in the downward facing direction, with an 8mm C-type lens protruding just below the bottom plate.

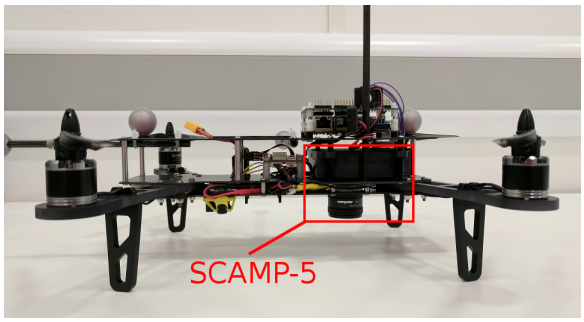

Fig. 2: Custom quadrotor used for experiments. SCAMP vision sensor integrated between top and bottom frame plates.

An ODROID XU4 single board Linux computer is fitted to the top of the quadrotor and enables the SCAMP and 'Pixhawk' autopilot to both be integrated within the Robot Operating System (ROS) for rapid development of the algorithms. No significant processing is carried out on the ODROID and CPU usage is minimal. Visual odometry data is passed from the SCAMP over a Serial Periphery Interface (SPI) link to the ODROID, meanwhile flight data from the Pixhawk is sent via a serial UART link. The roll and pitch angles of the aircraft from the flight data are passed on to the SCAMP system via an SPI link in order to perform image corrections, as will be described further in Section III-B. These communication links are summarised in Figure 3. The SCAMP-5 vision system has an M4 processor that could carry out the computations currently programmed on the ODROID and talk directly to the Pixhawk with the available serial link.

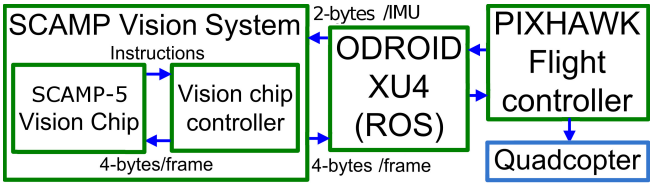

Fig. 3: Block diagram of hardware. ODROID is used for rapid development with ROS and passes data between flight controller and SCAMP vision system.

III. METHOD

A. Odometry

The SCAMP-5 PPA camera attached to the underside of the vehicle as shown in Figure 2 is used to produce an estimate of the vehicle's motion. The motion estimate is made

by building upon image alignment techniques first introduced in our previous paper [11]. The image alignment process used involves determining the transformation (consisting of translational, scaling and rotational components) to apply to the latest acquired image in-order to best align it with the current key-frame. This is determined by iteratively applying small transformations to the latest image, evaluating each to determine if it would result in an improved alignment with the key-frame, and rejecting those which do not. This process effectively performs a gradient descent search across possible image transformations, converging toward the transformation resulting in best local alignment with the key-frame. A motion model is also used to generate an initial estimate of the alignment transformation for each frame, reducing both the number of iterations required for convergence, and accordingly the computation time. Due to the high frame-rate capabilities of SCAMP-5, acquired images are typically free from motion blur and only exhibit small changes between frames. This allows robust image alignment to still be performed under rapid camera motion not possible when using a standard camera. A full description of this image alignment process is given in [11].

B. Perspective Correction

1) Overview: The image-alignment based estimation of the vehicle's motion is performed under the assumption that the SCAMP sensor is always orientated normal to the ground plane. However this assumption is clearly broken whenever the vehicle changes its pitch and roll angles in order to accelerate. Let θ denote the sensor's angle of deviation from being orientated normal to the ground plane, and angle α denote the sensor's field of view. The sensor then observes the area of ground plane lying within it's view frustum as shown in Figure 5. Visual features observed upon the ground plane will be at different distances from the sensor. This results in perspective distortion, under which the shape and appearance of such features will vary with the sensor's x, y position parallel to the plane. The magnitude of such distortion increases with angle θ, leading to situations where two images taken at different positions cannot be accurately aligned even if the same features are present in both. Thus the visual odometry is unable to function reliably whenever the vehicle undergoes rapid accelerations which vary angle θ.

To address this problem, perspective correction is applied to each image acquired by the SCAMP sensor. This process warps each image such that it appears as if it was taken at sensor angle $\theta = 0$ (facing normal to the ground plane) as illustrated by Figure 4. This perspective correction algorithm is performed entirely on-board the SCAMP sensor, using the vehicle's IMU data to determine angle θ and the correct warping to apply to the current image. This additional perspective correction step acts as a "virtual gimbal", producing images as though the sensor orientation was locked at $\theta = 0$, normal to the ground plane, allowing the visual odometry to continue to operate successfully during aggressive vehicle motion.

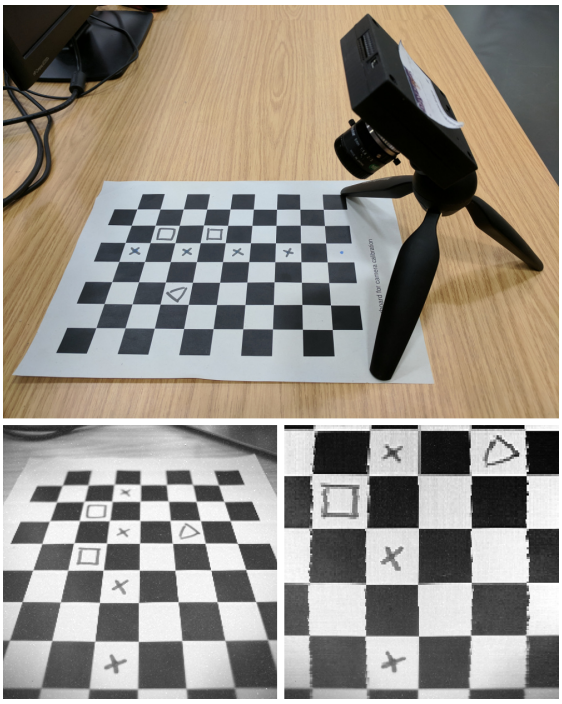

Fig. 4: An example of perspective correction. Top:SCAMP-5 observes an angled checkerboard Left: acquired image Right: image after the perspective correction has been applied.

2) Formulation: It is clear that the shape and size of the area of ground plane observed by the sensor changes and increases with angle θ as shown in Figure 5. Let W and L denote the center width and length of the observed area as denoted on Figure 5. The values of L and W change with angle θ according to Equations 1 and 2 respectively, which are plotted on Figure 6.

$$L = D\frac{2Sin(\alpha)}{Cos(2\theta + \alpha) + Cos(\alpha)} \tag{1}$$

$$W = D\frac{Sin(\alpha)}{Cos(\theta)Cos(\alpha)} \tag{2}$$

From this it is clear that the "length" L of the observed area of ground plane increases at a significantly greater rate than the "width" W, trending to infinity as θ approaches $\pi/2$ at which the sensor faces the horizon.

The sensor's view frustum is divided between the pixels of the sensor, each of which then observes some area of ground plane that determines its value in the final image. In this work we assume this division to be equal between pixels, with each pixel having a frustum of field of view $\beta = \alpha/256$, with 256 being both the horizontal and vertical resolution of the SCAMP sensor. This concept is illustrated in Figure 5, where for pixel(i, j), the angle of deviation is denoted $\theta_{i,j}$, the center length of its observed area by $l_{i,j}$, and the center width by $w_{i,j}$. We make the approximation that $\theta_{n,j} = \theta_{m,j} : \forall j$, and thus the angle of deviation for any pixel on the j^{th} row is given by Equation 3.

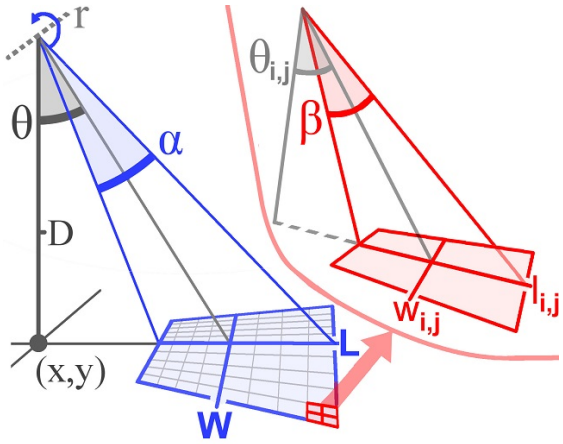

Fig. 5: Blue view frustum of sensor at angle θ, and field of view α, highlighting observed area of ground plane. This frustum of the sensor is divided equally among pixels. The frustum of a pixel(i, j) as shown in red has field of view β and associated angle $\theta_{i,j}$.

$$\theta_{i,j} = \theta_j = \theta + \beta(128 - j) \tag{3}$$

Under this approximation the area of ground plane observed by a pixel(i, j) varies with $\theta_{i,j}$ in the same manner as that observed by the sensor as a whole varies with θ, and thus $l_{i,j}$ and $w_{i,j}$ are defined by Equations 4 and 5.

$$l_{i,j} = l_j = D\frac{2Sin(\beta)}{Cos(2\theta_j + \beta) + Cos(\beta)} \tag{4}$$

$$w_{i,j} = w_j = D\frac{Sin(\beta)}{Cos(\theta_j)Cos(\beta)} \tag{5}$$

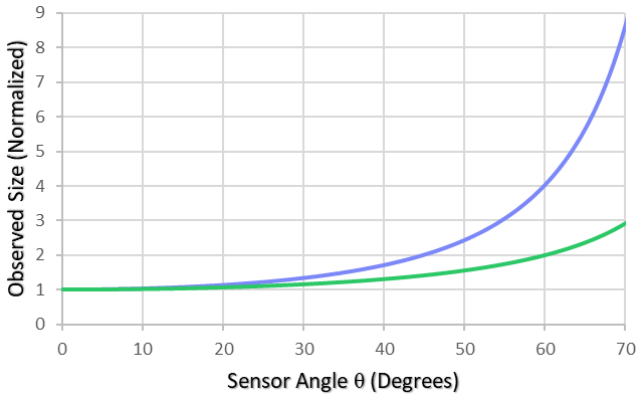

Fig. 6: A plot illustrating the change in the center length L (blue) and center width W (green) of the sensor's view frustum with increasing sensor angle θ relative to the ground plane. The field of view was taken to be $\alpha = 54.2°$ in accordance with the lens used in Section IV.

To apply perspective correction, the image is warped such that each pixel represents an observation of an area of ground plane of approximately equal size and shape. This is applied in two steps, first a vertical warping upon the columns of the image equalizing the length of the ground plane areas observed by each pixel, and a second horizontal warping upon rows equalizing the observed area widths. Note that before these warping are applied the image is rotated such that the projection of the axis of rotation for θ (denoted by r in Figure 5) into the image is parallel the rows of the image (i.e. parallel with the images x axis). This step is required to ensure that the horizontal and vertical warping applied to the image are correct no matter the axis of rotation about which the sensor is rotated by θ. After perspective correction has been applied to the rotated image the inverse rotation is applied to then obtain the final unrotated perspective corrected image. These image rotations are performed using the PPA rotation algorithm introduced in [11].

3) Vertical Warping: In order to create a perspective corrected image each pixel must represent an observation of an area of ground plane of approximately the same length. This is achieved by duplicating specific pixel rows within the image, effectively spreading the content of those rows vertically across multiple pixels. This results in an image where each pixel observes an area of ground plane of approximately the same length.

The locations at which these row duplications take place (along with the number of duplications to make at each location) is precomputed by the process listed in Algorithm 1. This involves iterating across a range of angles in increments of β, calculating the length of ground plane that would be observed by a pixel at each angle, and tracking the total length observed. At each iteration, the total observed length is compared to that expected for a perspective corrected image in which each pixel pixel(i, j) has angle $\theta_{i,j} = 0$. Wherever the difference between these two totals exceeds the length observed by a pixel(i, j) of angle $\theta_{i,j} = 0$, the appropriate number of rows are inserted to equalize the observed length per pixel.

This produces a lookup table which can then be efficiently used to determine where to insert duplicate rows into a given image acquired at a specific sensor angle θ, thus performing perspective correction along the image's vertical axis.

4) Horizontal Warping: The second warping adjusts the rows of the image such that the width of the area of ground plane observed by each pixel is approximately equal. Ideally this would involve inserting duplicate pixels within each row at specific locations, however this is a costly operation to perform upon every row of the image. Instead we settle for an approximation whereby each row of the image is rescaled horizontally. This approximation still produces perspective corrected images of the quality seen in Figure 4, which are adequate for our purposes. In a similar manner to the vertical warping process, the locations at which pixel rows are upscaled along with the magnitudes of these rescaling are precomputed using the process listed in Algorithm 2, and stored in a lookup table.

Similar to Algorithm 1, Algorithm 2 iterates across a range of angles, in increments of β. For each angle, the total width

Algorithm 1 $Precompute\_vertical\_PC(\beta, N)$

INPUT :
β *// FOV of pixel frustum*
N *// Range to precomputed*
OUTPUT :
$Locations$ *// row duplication locations*
$Magnitudes$ *// row duplication amounts*

$Default\_area = \frac{Sin(\beta)}{Cos(\beta)}$
$Area\_Cntr = 0$
for $n = 0$ to N **do**
  $Pix\_angle = n \times \beta$
  $Pix\_area = \frac{2Sin(\beta)}{Cos(2 \times Pix\_angle + \beta) + Cos(\beta)}$
  $Duplicated\_rows = 0$
  $Area\_Cntr = Area\_Cntr + Pix\_area$
  $Area\_Cntr = Area\_Cntr - Default\_area$
  while $Area\_Cntr > Default\_area$ **do**
    $Duplicated\_rows = Duplicated\_rows + 1$
    $Area\_Cntr = Area\_Cntr - Default\_area$
  end while
  if $Duplicate\_rows > 0$ **then**
    Add n to $Locations$
    Add $Duplicated\_rows$ to $Magnitudes$
  end if
end for
return $Locations, Magnitudes$

Algorithm 2 $Precompute\_horizontal\_PC(\beta, N)$

INPUT :
β *// FOV of pixel frustums*
N *// Range to precomputed*
OUTPUT :
$Locations$ *// row scaling change locations*
$Magnitudes$ *// row scaling change values*

$Prev\_scaling = 0$
$Default\_area = 2 * tan(\beta 128)$
for $n = 0$ to N **do**
  $Row\_angle = n\beta$
  $Row\_area = \frac{2 * tan(\beta 128)}{Cos(Row\_angle)}$
  $Area\_ratio = \frac{Row\_area}{Default\_area}$
  $Row\_scaling = Round(255(Area\_ratio - 1))$
  if $Row\_scaling > Prev\_scaling$ **then**
    Add n to $Locations$
    $Scaling\_change = Row\_scaling - Prev\_scaling$
    Add $Scaling\_change$ to $Magnitudes$
    $Prev\_scaling = Row\_scaling$
  end if
end for
return $Locations, Magnitudes$

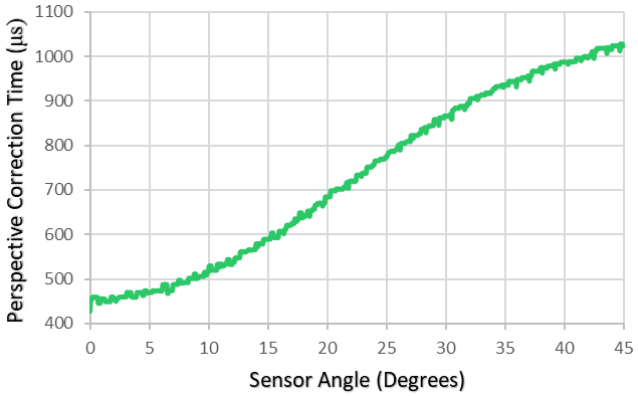

Fig. 7: A plot of how the computation time of the perspective correction process varies with the angle of the sensor which needs to be corrected.

of the area of ground plane observed by a row of pixels at said angle is calculated and compared to the width which should be observed in a perspective corrected image. The upscaling factor that should be applied to the current row of pixels in order to apply perspective correction and equalize the area of ground plane observed across the row's pixels is then calculated. Note that due to the nature of SCAMP and the employed method of image scaling as described in [11], this upscaling factor is rounded to discrete values. Whenever this rounded scaling factor varies from that calculated for the row in the previous iteration, the location and magnitude of this change scaling are recorded. This process produces a lookup table which is used to determine the horizontal scaling to apply to each row of an image, acquired at sensor angle α, in order to apply perspective correction. In practice, entire horizontal slices of the image are horizontally scaled at once, rather than scaling on a per row basis, in order to improve efficiency.

5) Computation Time: Figure 7 shows the how the computation time of the entire perspective correction process varies with sensor angle Θ, taking the sensor's field of view to be $54.2°$ degrees to match the lens used in the experiments presented in Section IV. As to be expected computation time increases with angle, but still remains under a millisecond up to around an angle of 45 degrees.

C. High Dynamic Range Edge Detection

An HDR edge detection algorithm was used to produce the images for the visual odometry algorithm. This process involves obtaining a sequence of images, each of longer exposure length than the last. Edge detection is performed upon each, thus detecting edges visible at various exposure lengths. The edges detected at each exposure length are then combined, forming a single image containing edges detected both in dark and bright regions of the scene. Figure 8 illustrates this process outlined in Algorithm 3.

This HDR edge detection is vital in allowing the visual odometry algorithm to successfully operate across various lighting conditions.

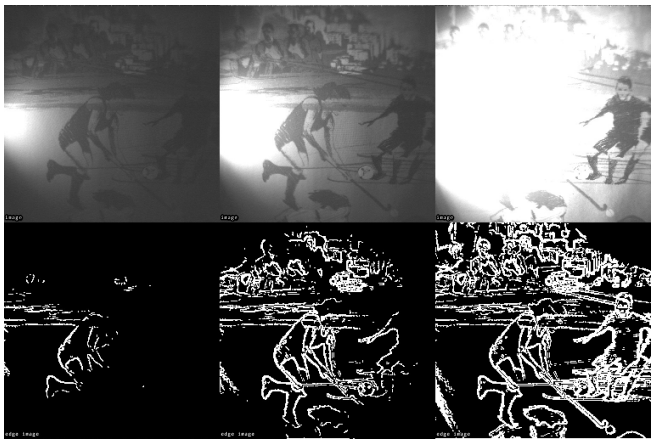

Fig. 8: An example of HDR edge extraction on SCAMP-5. The sensors image is increasingly exposed from left to right. With each exposure edges are extracted, all of which are accumulated to form the final edge image.

Algorithm 3 HDR edge detection(N,T,E)

INPUT:
N *// HDR Iterations*
T *// exposure time (μs) per iteration*
E *// edge detection threshold*

Clear(R5,R6) *// clear binary maps*
for $n = 0$ to N **do**
  Sleep(T) *// further expose current sensor image*
  Get_Image(A) *// copy image to analog array A*
  // extract edges from A into binary map R5
  Extract_Edges(R5,A,E)
  R6 = OR(R6,R5) *// merge extracted edges*
end for
return R6

D. Integration

The SCAMP is programmed to output odometry information in the form of pixels shifted per frame for roll and pitch in the aircraft frame. Rotation in the aircraft's yaw is given in the form of the number of rotation steps per frames in increments of $0.00451°$ or 0.0000787 radians. These values can be turned into rotational rates and then in turn integrated to find vehicle heading. These computations would be possible on the SCAMP, but are computationally trivial and performed on the ODROID for convenience.

The SCAMP sensor has 256x256 pixels and for these experiments was fitted with a $54.2°$ horizontal/vertical FoV lens (or 0.9146 radians). Distance of the quadrotor above ground is found using the attached height sensor (TeraRanger One) and sent to the ODROID from the Pixhawk. Rotational rates in roll ($\dot{\phi}_{vo}$) and pitch ($\dot{\theta}_{vo}$) due to visual flow about the SCAMP may then be calculated as follows:

$$\dot{\phi}_{vo} = \left(\frac{dy}{256} \times 0.9146\right) \times \frac{1}{dt} \quad (6)$$

$$\dot{\theta}_{vo} = \left(\frac{dx}{256} \times 0.9146\right) \times \frac{1}{dt} \quad (7)$$

in radians per second, where dx and dy are the number of pixels the image has shifted in the x and y directions and dt is time since the last frame was returned. The angular velocity of the aircraft may then be subtracted from this rotational rate to determine the component due to translation of the aircraft, which in turn is multiplied by distance to the ground plane to give translational velocity,

$$v_x = \left(\dot{\theta}_{vo} - \dot{\theta}\right) \times h \quad (8)$$

$$v_y = \left(\dot{\phi}_{vo} - \dot{\phi}\right) \times h \quad (9)$$

where v_x is the aircraft's forward velocity and v_y is its velocity to the right. The aircraft's roll and pitch rates are $\dot{\phi}$ and $\dot{\theta}$ respectively.

The yaw rate of the aircraft may be taken from the rate of yaw of the SCAMP,

$$\dot{\psi}_{vo} = dz \times \frac{0.0000787}{dt} \quad (10)$$

where $\dot{\psi}_{vo}$ is the yaw rate due to visual odometry in radians per second and dz is the number of rotation steps given by the SCAMP.

The two velocity components and yaw rate may all be integrated with time to calculate displacement over the course of the flight. Velocity information should be filtered due to the resolution of matches, *e.g.* one pixel shift at 500 fps due to pure translation at one metre altitude would equate to $1.8m/s$. A low pass filter with the pole located at -30 was applied to smooth out the velocity data. The algorithm performs five search iterations for image alignment each frame; this leads to a maximum measurable angular velocity of $512°/s$ or a translation of $8.9m/s$ at one metre above ground. As height increases the velocity at which tracking may continue also increases.

IV. RESULTS

This section highlights some of the results captured during testing, demonstrating the quality of the position and velocity information provided from the visual odometry. Ground truth measurements from a motion capture system were taken for comparison.

A. Velocity Estimation

The quadrotor was commanded to fly fore and aft at speeds reaching just under $4m/s$ and pitch angles of around $25°$. A Vicon motion capture system logged the quadrotor position over time allowing the velocity estimates from the visual odometry to be compared against a ground truth. Figure 9 shows how the velocity from visual odometry closely matches the vicon data. The accuracy of the result is impressive given both the speed of the aircraft at only one metre above the ground as well as the large pitch angle deviations undertaken. The reader is directed to the attached video for a demonstration of these flights and the behaviour of the perspective correction that helps make this possible.

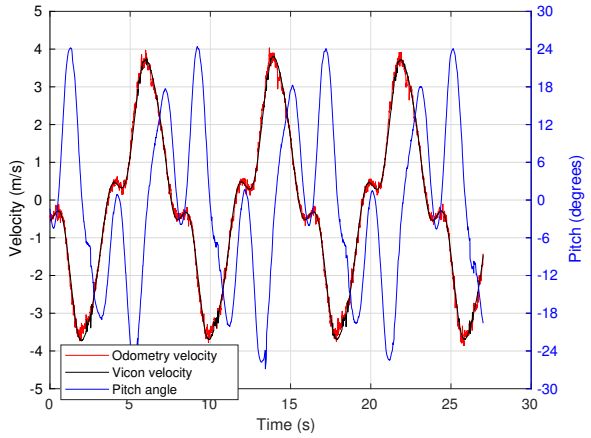

Fig. 9: Quadrotor flying forwards and backwards at up to four metres per second at around 1m above ground. Comparing visual odometry with ground truth from mocap

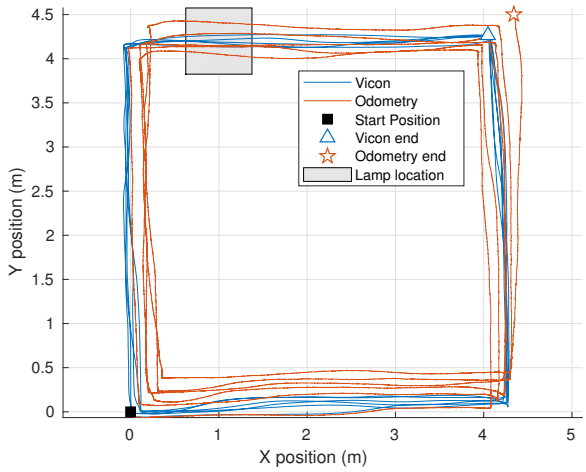

Fig. 10: Quadrotor flying square pattern. Comparing visual odometry with ground truth

B. Position Drift

Two tests were performed to test the accuracy and drift of the integrated visual odometry. The first, was a hover on the spot test for three minutes with the difference between Vicon and the visual odometry recorded. During the three minute hover at 1.2m above ground, the difference between the two measurements had a maximum of 8.2cm with a standard deviation of 1.6cm. The odometry data should therefore be sufficient for holding position during long periods of hovering indoors.

The performance of the integrated visual odometry was also compared whilst flying around a 4x4 metre square trajectory at 1 metre above the ground. The resulting path is shown in Figure 10, which was 100m long and had a final drift of 0.39m between the estimated position and Vicon reported position. The magnitude of final drift over path length compares well with the PX4Flow device, which was tested in a similar way [6] and found to have a final drift of 25cm after 28.44m flight (or 0.8% of path length) versus the result presented here having 39cm drift after 100m of flight (*i.e.* 0.4% of the path length). Table I summarises the results from the hover and square route tests, where errors are taken as the distance between Vicon measured position and the estimated position from visual odometry, compared every 10ms.

Test	Std. Dev.	Max error
3min Hover	1.6cm	8.2cm
100m Square Pattern	11cm	47cm

TABLE I: Errors in position estimates flying indoors

C. Yaw Estimation

The visual odometry can also estimate the heading or yaw angle, which could help when magnetic interference interferes with the autopilot's estimate of heading. A similar trajectory to that in Figure 10 was flown, but with the quadrotor commanded to yaw in the corners. The resulting time history of the yaw estimates from the odometry and autopilot are compared against the Vicon measurements in Figure 11. It can be seen that the odometry follows the vicon measurement of yaw fairly well, with a slight under estimate of turn three causing an offset from fifty seconds onwards; the estimate is, however, better than the autopilot for the duration of the flight.

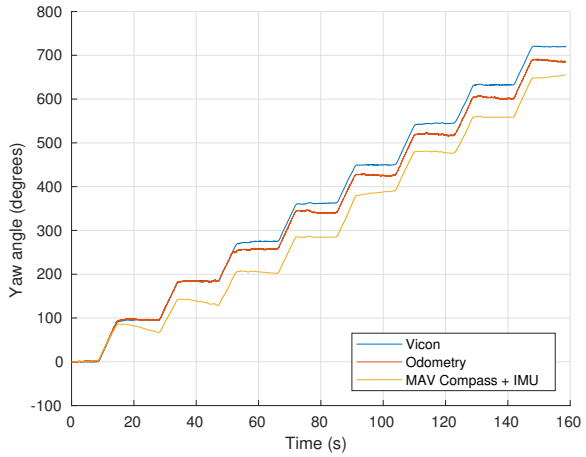

Fig. 11: Quadrotor following a 4x4 metre square indoors, yawing at the corners. Comparing visual odometry with ground truth and autopilot's estimates of yaw angle.

D. High Dynamic Range

A bright flood lamp was placed upon the floor whilst the MAV flew the square path shown in Figure 10, located in the top left corner of the trace. The HDR algorithm as described in Section III-C was able to use a range of exposures to capture detailed edges rather than suffering from over exposure as shown by the photograph in Figure 12. Note, the edge image displayed on the right is not normally output by the SCAMP during flight, but it was reprogrammed to generate this figure and illustrate the effect of the HDR method. Due to this handling of HDR scenes, the visual odometry result

was not perturbed by the presence of the lamp, which cannot be said for the PX4Flow that was attached at the same time. Figure 13 shows the visual odometry trace calculated from flow data returned by the PX4Flow during the same flight, which reports noisy data when flying over the lamp.

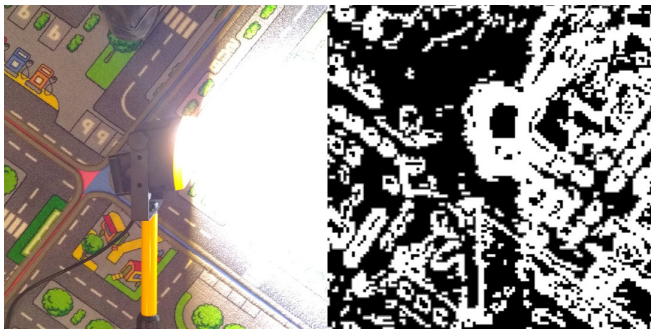

Fig. 12: Left: Photograph of lamp laid along path of route flow in Fig. 10. Right: Edges detected by SCAMP-5, utilising HDR algorithm.

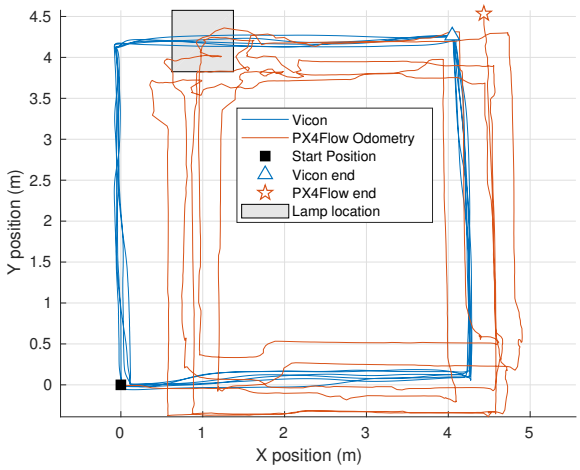

Fig. 13: PX4Flow visual odometry path data, same flight as Figure 10. PX4Flow struggles when flying over lamp placed in top left corner

V. CONCLUSION

This work has shown that a novel Pixel Processor Array device could significantly aid a micro air vehicle in navigating within an indoor environment. The velocity results demonstrate excellent correlation with motion capture ground truth, despite high velocities and large aircraft angles. The tracking results show that drift over a path flown indoors is better than the dedicated PX4Flow sensor, but with the added benefit of also tracking yaw, handling HDR scenes and operating at up to double the frame rate.

In addition to the quality of the tracking, the PPA device brings many features that can improve the robustness of the system. The high frame rate that very few other sensors can achieve means rapid manoeuvring is possible. High dynamic range means that the vision sensor can perform in a wide variety of demanding lighting conditions that might otherwise saturate other systems. It is also possible to reprogram the PPA device for other tasks, such as target tracking [14], with no need to carry additional computer hardware for visual processing. Given this flexibility, future work could investigate the image matching techniques used here for localisation.

Data Access and Acknowledgements Supported by UK EPSRC EP/M019454/1 and EP/M019284/1. The nature of the PPA means that the data used for evaluation in this work is never recorded.

REFERENCES

[1] S. Shen, Y. Mulgaonkar, N. Michael, and V. Kumar, "Vision-based state estimation and trajectory control towards high-speed flight with a quadrotor." in *Robotics: Science and Systems*, vol. 1. Berlin, Germany, 2013.

[2] M. Bloesch, S. Omari, M. Hutter, and R. Siegwart, "Robust visual inertial odometry using a direct ekf-based approach," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE, 2015, pp. 298–304.

[3] C. Fu, A. Carrio, and P. Campoy, "Efficient visual odometry and mapping for unmanned aerial vehicle using arm-based stereo vision pre-processing system," in *Unmanned Aircraft Systems (ICUAS), 2015 International Conference on*. IEEE, 2015, pp. 957–962.

[4] R. Strydom, S. Thurrowgood, and M. V. Srinivasan, "Visual odometry: autonomous uav navigation using optic flow and stereo," in *Australasian Conference on Robotics and Automation (ACRA)*. Australian Robotics and Automation Association, 2014, pp. 1–10.

[5] R. G. Valenti, I. Dryanovski, C. Jaramillo, D. P. Ström, and J. Xiao, "Autonomous quadrotor flight using onboard rgb-d visual odometry," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 2014, pp. 5233–5238.

[6] D. Honegger, L. Meier, P. Tanskanen, and M. Pollefeys, "An open source and open hardware embedded metric optical flow cmos camera for indoor and outdoor applications," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 1736–1741.

[7] P. Li, M. Garratt, A. Lambert, M. Pickering, and J. Mitchell, "Onboard hover control of a quadrotor using template matching and optic flow," in *Proceedings of the International Conference on Image Processing, Computer Vision, and Pattern Recognition (IPCV)*. The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), 2013, p. 1.

[8] V. Grabe, H. H. Bülthoff, D. Scaramuzza, and P. R. Giordano, "Nonlinear ego-motion estimation from optical flow for online control of a quadrotor uav," *The International Journal of Robotics Research*, vol. 34, no. 8, pp. 1114–1135, 2015.

[9] A. Lopich and P. Dudek, "A SIMD cellular processor array vision chip with asynchronous processing capabilities," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 58, no. 10, pp. 2420–2431, 2011.

[10] S. J. Carey, A. Lopich, D. R. Barr, B. Wang, and P. Dudek, "A 100,000 fps vision sensor with embedded 535gops/w 256× 256 simd processor array," in *VLSI Circuits (VLSIC), 2013 Symposium on*. IEEE, 2013, pp. C182–C183.

[11] L. Bose, J. Chen, S. J. Carey, P. Dudek, and W. Mayol-Cuevas, "Visual odometry for pixel processor arrays," in *International Conference on Computer Vision (ICCV), 2017 IEEE International Conference on*. IEEE, 2017.

[12] J. N. Martel, L. K. Müller, S. J. Carey, and P. Dudek, "Parallel hdr tone mapping and auto-focus on a cellular processor array vision chip," in *Circuits and Systems (ISCAS), 2016 IEEE International Symposium on*. IEEE, 2016, pp. 1430–1433.

[13] J. N. Martel, L. K. Mueller, S. J. Carey, and P. Dudek, "A real-time high dynamic range vision system with tone mapping for automotive applications," *CNNA 2016*, 2016.

[14] C. Greatwood, L. Bose, W. Richardson, T. S.and Mayol-Cuevas, J. Chen, S. J. Carey, and P. Dudek, "Tracking control of a uav with a parallel visual processor," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2017.